When Sentiment Goes Sideways: Failure Modes, Negation Scope, Sarcasm, and Domain Drift in "Meta-Style" PyTorch Sentiment for Ordinary Human Text (Social, Support, Reviews, Email)

Douglas C. Youvan

doug@youvan.com

www.youvan.ai

September 29, 2025

Off-the-shelf "Meta-style" sentiment stacks—fast PyTorch baselines with optional transformer heads—perform impressively on benchmarks, yet stumble in the wild where ordinary humans write like, well, humans. Polarity flips hide in short clauses ("not bad at all"), irony signals are subtle ("great, just what I needed…"), and domain drift turns harmless tokens into red flags ("sick" = praise in gaming slang; "killer" = positive in music reviews). Lexicon seeds overweight obvious cues while missing pragmatic ones; transformers inherit training biases, struggle with code-switching and dialects, and degrade under distribution shift (support tickets vs tweets vs clinical notes). We systematize these **failure modes** and pair them with **diagnostic plots** that make mistakes legible: positive/negative/net tracks, negation density, sarcasm proxies, uncertainty bands, and genre detectors. We also propose **cheap fixes** that don't require giant re-training runs: windowed negation rules, sarcasm heuristics, small domain lexicons, label smoothing, temperature-scaled calibration, and shallow genre gating (Complaint, Praise, Irony, Instruction). The goal is an operations-ready recipe: keep the PyTorch simplicity and speed, add lightweight guardrails, and expose interpretable dashboards so analysts can see *why* a score moved. With these upgrades, sentiment becomes a trustworthy signal for social monitoring, customer support, product reviews, and internal email—without pretending language is always literal.

Keywords: sentiment analysis, PyTorch, Meta-style baseline, transformer diagnostics, negation scope, sarcasm detection, domain adaptation, code-switching, dialect bias, uncertainty calibration, genre classification, lexicon augmentation, social media, customer support, product reviews, email analytics.

A collaboration with GPT-5. CC4.0. 42 pages.

Outline for Readers

1. **Why Ordinary Text Breaks Benchmarks**
    - Literal labels vs pragmatic meaning; distribution shift in the wild
2. **Catalog of Failure Modes**
    - Negation scope & intensifiers (“not bad”, “hardly great”, “sooo good”)
    - Sarcasm/irony & understatement (“perfect…”)
    - Domain drift & slang (“wicked”, “sick”, “fire”)
    - Code-switching, AAVE, and dialect bias
    - Emojis, hashtags, punctuation style, ALL CAPS
    - Quoted speech vs speaker stance; multi-speaker threads
    - Entity leakage (brand names, “crash”, “killer feature”)
    - Length and list effects; template boilerplate in support emails
3. **Methods (Reproducible, Light-Weight)**
    - PyTorch EmbeddingBag baseline + optional transformer head
    - Multi-line diagnostics: pos/neg/net, negation density, sarcasm proxy, length norm
    - Confidence & calibration: temperature scaling, entropy bands
4. **Case Studies (Small, Realistic)**
    - Social posts (slang + emojis)
    - Support tickets (mixed praise/problem)
    - Product reviews (irony, hedged praise)
    - Workplace email (polite negativity, faint praise)
5. **Mitigations Without Massive Training**
    - Windowed negation rules; intensifier tables
    - Tiny domain lexicons (per team or product)
    - Genre gating (Complaint/Praise/Irony/Instruction) before sentiment

- Sarcasm heuristics (contrastive cues, punctuation, emoji sets)
- Calibration and abstention (“low confidence” routing)

6. **Dashboards for Analysts**
    - Multi-line plots, delta views, drift monitors, error notebooks
7. **Evaluation & Monitoring**
    - Balanced test slices; human review loops; regression alarms
8. **Ethics & Risk**
    - Dialect fairness, workplace impact, moderation sensitivity
9. **Limitations & Future Work**
    - Beyond heuristics; weak supervision; small fine-tunes; multimodal cues
10. **Reproducibility Appendix**

    Minimal PyTorch scripts, lexicon seeds, negation rules, calibration tips

References

Art

1) Why Ordinary Text Breaks Benchmarks

Literal labels vs pragmatic meaning; distribution shift in the wild

1.1 Benchmarks are literal; people are pragmatic.
Most benchmark items anchor a single, context-free polarity, but ordinary writers hedge, hint, and negotiate face. The same surface string can invert meaning with micro-cues: "not bad at all" (positive via negation+intensifier), "great… 🙃" (sarcasm via emoji), "Thanks for nothing" (idiomatically negative despite *thanks*). Literal labels reward token spotting; pragmatic intent requires discourse cues (speaker goals, stance, audience, stakes).

1.2 Pragmatic operators the model underweights.

- **Negation scope & ladders:** not, hardly, scarcely, barely, no longer (different strengths).
- **Hedging & boosters:** kind of, a bit, maybe vs very, so, unbelievably.
- **Politeness masks:** "Could you please…" can wrap harsh criticism.
- **Irony markers:** contrastive conjunctions ("yeah, right"), scare quotes, ellipses, stretched spellings ("greeeaaat"), mismatch of positive words with negative events.
- **Community slang:** sick, fire, wicked (domain-specific positives).

1.3 Distribution shift is the default, not the exception.
Training corpora ≠ deployment streams. Social → Support → Email → Reviews each has distinct priors: length, punctuation, emoji density, register, domain terms. Shift types:

- **Covariate shift:** p(x) changes (emoji/hashtag frequency, product jargon).
- **Label shift:** p(y) changes (support tickets skew negative by design).
- **Concept drift:** mapping x → y evolves (sick becomes positive in a community, then fades).
- **Speaker mix drift:** more non-native, code-switching, or AAVE—models misread grammar as sentiment.

1.4 Why "good" models still fail.

- **Spurious lexical shortcuts:** model keys on *great/awful* and ignores negation or irony.
- **Length & list effects:** long tickets mix praise and complaints; verse-like lists flip polarity clause by clause.
- **Quoting and stance:** "He said 'this product is terrible'" is reportage, not endorsement.

- **Annotation bias:** crowdworkers label literal polarity; pragmatic intent unlabeled → models overfit the literal.
- **Calibration errors:** confident wrong answers under shift; softmax ≠ probability.

1.5 Practical checks before you trust a score.

- **Negation window test:** inject "not/no longer" around positive spans; polarity should flip with controlled magnitude.
- **Sarcasm probe set:** positive tokens + negative outcomes ("Fantastic, another outage")—expect neutral/negative with high uncertainty.
- **Domain lexicon audit:** sample top tokens driving decisions; add/flag slang and brand terms.
- **Genre gate first:** classify Complaint/Praise/Irony/Instruction, then apply genre-aware sentiment rules.
- **Confidence governance:** use temperature scaling; abstain/route when entropy is high or drift monitors trigger.

1.6 Takeaway.
Benchmarks certify *literal* competence; production requires *pragmatic* competence under shift. Treat sentiment as a measured signal with diagnostics, not an oracle.

2) Catalog of Failure Modes

2.1 Negation scope & intensifiers ("not bad", "hardly great", "sooo good")

- **Symptom:** Polarity reads literal (bad/ great/ good) and ignores "not/ hardly/ scarcely," or overreacts to boosters ("sooo").
- **Why it fails:** Token-level shortcuts; no explicit scope modeling; intensifier ladders unmodeled.
- **Quick tests:** Flip-window test ("not ___"), ladder test (a bit → very → unbelievably).
- **Mitigations:** Windowed negation rules; dependency-based shifters; calibrated intensifier table; treat elongated spellings as normalized features.

2.2 Sarcasm/irony & understatement ("perfect…")

- **Symptom:** Positive tokens in negative contexts score positive ("Great, another outage").
- **Why it fails:** Sarcasm cues are indirect: contrast, ellipses, scare quotes, mismatch between sentiment and event.
- **Quick tests:** Contrast probes: positive adjective + negative outcome; expect ≤ neutral with high uncertainty.
- **Mitigations:** Sarcasm heuristics (ellipsis, scare quotes, exclamation + complaint); add a light sarcasm head; raise abstain threshold under mismatch.

2.3 Domain drift & slang ("wicked", "sick", "fire")

- **Symptom:** Community praise read as negative or neutral; time-varying meanings.
- **Why it fails:** Training domains differ; slang polysemy shifts by community and year.
- **Quick tests:** Small slang lists per domain; A/B scores across time slices.
- **Mitigations:** Tiny domain lexicons; per-stream calibration; periodic drift audits and lexicon refresh.

2.4 Code-switching, AAVE, and dialect bias

- **Symptom:** Neutral/positive dialectal forms misread as negative or "toxic"; grammar ≠ sentiment.
- **Why it fails:** Underrepresentation in training; spurious correlations between form and label.
- **Quick tests:** Dialect-balanced evaluation sets; compare scores across paraphrases.
- **Mitigations:** Genre/dialect gate before sentiment; fairness constraints; targeted augmentation; human review for flagged cohorts.

2.5 Emojis, hashtags, punctuation style, ALL CAPS

- **Symptom:** Emojis dominate polarity; hashtags flip meaning (#blessed vs #fail); !!! or ALL CAPS over-amplify.
- **Why it fails:** Non-word signals out of distribution; punctuation treated as intensity, not discourse.
- **Quick tests:** Remove vs keep emojis/hashtags; measure delta; intensity-sensitivity curve for ! and CAPS.

- **Mitigations:** Emoji/hashtag lexicons with sense disambiguation; cap intensity effects; treat repeated punctuation as bounded features.

2.6 Quoted speech vs speaker stance; multi-speaker threads

- **Symptom:** Reported negativity ("He said 'terrible'") attributed to the writer; reply inherits prior speaker's sentiment.
- **Why it fails:** No speech-attribution or thread structure; quotes parsed as author stance.
- **Quick tests:** Quote-tag probes; thread turn-taking probes.
- **Mitigations:** Quote detection + stance rules; thread-aware models (speaker turns); down-weight quoted polarity unless affirmed.

2.7 Entity leakage (brand names, "crash", "killer feature")

- **Symptom:** Words with technical senses read as emotional; brand names become polarity shortcuts.
- **Why it fails:** Shortcut learning: entity co-occurs with sentiment in training.
- **Quick tests:** Mask entities; compare scores (with vs [MASK]).
- **Mitigations:** Entity masking during inference; domain-specific sense lists (crash=bug, neutral); counterfactual data augmentation.

2.8 Length and list effects; template boilerplate in support emails

- **Symptom:** Long messages average out; list items flip polarity clause-by-clause; templates (greetings, signatures) dilute signal.
- **Why it fails:** Bagging over heterogeneous spans; boilerplate overwhelms content.
- **Quick tests:** Segment-level scoring (bullet/paragraph); compare raw vs boilerplate-stripped.
- **Mitigations:** Segment and aggregate (min/median or weighted by complaint cues); boilerplate filters; per-section sentiment + final fusion.

2.9 Putting it to work (ops checklist)

- Add **negation/intensifier** tables and windowed scope.
- Gate by **genre/dialect** before sentiment.
- Maintain **domain lexicons** (slang, technical senses, emojis, hashtags).

- Use **entity masking** and **counterfactual tests**.
- Score **by segment**, then aggregate; strip boilerplate.
- Track **drift**, **calibrate** (temperature), and **abstain** when uncertainty is high.

3) Methods (Reproducible, Light-Weight)

3.1 Data & preprocessing (one file, 1..N rows)

- Input: a plain UTF-8 text file with one document per line (verse, tweet, ticket, review, email paragraph).
- Tokenization: lowercase; split on [^A-Za-z]+; drop empties.
- Optional stopword handling: keep function words (they carry negation scope); do not strip not, no, never, etc.
- Length features: tokens per line L; use len_norm = 1/sqrt(L).

3.2 PyTorch EmbeddingBag baseline (transparent, fast)

- Vocab: build from training split only; OOV -> id 0 with weight 0.
- Two scalar embeddings (dim=1): E_pos, E_neg with shape [V,1].
    - Initialize from small lexicons (domain seeds). Unknowns = 0.
    - Inference score per doc:
        - pos_sum = EmbeddingBag_sum(E_pos, doc_tokens)
        - neg_sum = EmbeddingBag_sum(E_neg, doc_tokens)
        - net = (pos_sum + neg_sum) * len_norm
- Optional fine-tune: freeze 80% of entries; allow updates for a small, curated subset (slang, domain terms) with L2=1e-4 and tiny LR (1e-3) for <= 3 epochs on a hand-labeled set (hundreds, not thousands).

3.3 Optional transformer head (drop-in, offline)

- Local HF folder (e.g., RoBERTa/RoBERTa-mini) fine-tuned for 2–3-class sentiment.
- Tokenize with local tokenizer; truncate to 256 tokens.
- Output: logits z in R^K; probabilities p = softmax(z/T) after calibration (Section 3.6).

- Keep both baselines: report EmbeddingBag scores and transformer probabilities side-by-side.

3.4 Negation scope (windowed shifter, no parser required)

- Negators: {not, no, never, n't, hardly, scarcely, barely, no_longer}.
- Window rule: within a right window of W tokens (default W=4), invert the polarity contribution of the first eligible sentiment token; stop at punctuation .?!;: or coordinating but.
- Implementation:
    - Scan tokens left->right; when negator seen, set flip_budget=1 and flip_until = min(i+W, next_punct_or_but).
    - For each sentiment token j with i < j <= flip_until, apply score *= -1 and set flip_budget=0.
- Intensifiers/attenuators: table of multipliers, e.g., {"very":1.5,"so":1.4,"really":1.3,"slightly":0.7,"somewhat":0.8} applied to the **next** sentiment token inside a window of 2.

3.5 Sarcasm proxy (cheap heuristics)

- Binary/real-valued features per doc:
    - Positive word + negative event collision, e.g., (pos_adj & outage|delay|crash|refund|complaint).
    - Ellipses or scare quotes present: ..., "great".
    - Contrastive conjunction after praise within 5 tokens: but, though, yeah right.
    - Punctuation intensity: ! count (capped at 3), stretched spellings (sooo, greeeaaat).
- Combine into sarcasm_proxy in [0,1] via min-max scaling of the summed cues; use as:
    - a plotted diagnostic line, and
    - a down-weight on final polarity: net_adj = net * (1 - 0.5*sarcasm_proxy).

3.6 Length normalization

- Default: net = (pos_sum + neg_sum) / sqrt(L).
- Also compute z-scores for plotting comparability across streams:

- z(x) = (x - mean(x)) / std(x) on the evaluation slice (do not leak across time spans).

3.7 Multi-line diagnostics (what we plot)

- Lines per x=indexed item:
    1. pos_sum * len_norm
    2. neg_sum * len_norm
    3. net
    4. negation_density = (#negators)/L
    5. sarcasm_proxy (0..1)
    - Optional: token count L as a faint reference or secondary axis.
    - Optional: transformer p_pos or p_neg as an additional line.
- Also provide delta view: |Delta(line)| to visualize responsiveness.

3.8 Confidence & calibration (simple, reliable)

- **Temperature scaling** (validation set):
    - Find scalar T>0 minimizing NLL on held-out predictions of the transformer head.
    - Use softmax(z/T) at inference; store T per domain/stream.
- **Entropy bands** (uncertainty gate):
    - Entropy H(p) = -sum_k p_k * log p_k; normalize by log K to H_norm in [0,1].
    - Define thresholds:
        - H_norm <= 0.3: high confidence → auto-action allowed.
        - 0.3 < H_norm <= 0.6: medium → log and show to analyst.
        - H_norm > 0.6: abstain/route to human or request more context.
- **Agreement check**:
    - If transformer sign disagrees with EmbeddingBag sign and H_norm > 0.5, flag for review.

- **Drift guard**:
    - Track running mean/std of negation_density, sarcasm_proxy, and token length. If any exceed 3*std from a 30-day baseline, raise a drift alert and freeze auto-actions.

3.9 Minimal file outputs (for reproducibility)

- scores.csv: per item: id, pos, neg, net, negation_density, sarcasm_proxy, len, (p_pos,p_neu,p_neg), H_norm.
- diagnostics.png: multi-line plot (one chart, no explicit color settings).
- deltas.png: absolute per-item change plot.
- calib.json: temperature T, thresholds, window sizes, lexicon version stamp.

3.10 Default hyperparameters (good first pass)

- Negation window W=4; intensifier window=2.
- Sarcasm weights: collision=1.0, contrastive=0.8, ellipses/scare quotes=0.6, punctuation/stretch=0.4; min-max to [0,1].
- EmbeddingBag LR=1e-3 (if fine-tuning selected), epochs<=3, batch=64.
- Transformer max_len=256; temperature initialized at 1.0; calibrated per stream.

3.11 Evaluation protocol (light, honest)

- Balanced slice per domain (e.g., 200 items: 50 complaint, 50 praise, 50 ironic, 50 neutral/instruction).
- Report: macro F1, ECE (expected calibration error), abstention rate, and disagreement rate (baseline vs transformer).
- Keep a small, rotating golden set for monthly drift checks.

This stack keeps the **PyTorch simplicity** while adding the three things production needs: scope-aware rules, interpretable diagnostics, and calibrated confidence with abstention.

4) Case Studies (Small, Realistic)

4.1 Social posts (slang + emojis)

Text A: “that drop was **sick** 🔥 🔥 could’ve done without the queue tho…”

- Signals: slang positives (sick, fire), mild complaint tail (“queue”). Emojis reinforce praise; “tho” introduces soft contrast.
- Baseline pitfall: flags “queue” → negative, underweights slang → net skewed low/neutral.
- Diagnostics: high love/pos track early, small neg bump at tail; sarcasm proxy low; length norm stable.
- Fix: domain lexicon {sick, fire}=positive; contrast rule keeps overall positive; output: positive with “minor friction” tag.

Text B: “amazing… another crash 😳”

- Signals: positive token with ellipsis + negative event + ironic emoji.
- Baseline pitfall: reads “amazing” as positive.
- Diagnostics: sarcasm proxy high (ellipsis + contrast + emoji), negation density low; net should be reduced/negative.
- Fix: sarcasm down-weight + event-collision rule (pos word + crash) ⇒ negative, high-uncertainty band.

4.2 Support tickets (mixed praise/problem)

Text C: “Love the new UI, but the **export keeps failing** for CSV.”

- Signals: praise + specific failure. Clause order matters.
- Baseline pitfall: average to neutral; actionability lost.
- Diagnostics: two peaks (pos, then neg); delta plot shows sharp drop at “failing.”
- Fix: segment-and-aggregate: label segments (Praise, Problem). Final: “Overall negative for task success; preserve praise snippet for product team.” Route with entity mask: export/CSV to the right queue.

Text D: “Thanks in advance, this **still doesn’t work** on 1.2.4.”

- Signals: politeness wrapper + hard negation + version marker.
- Baseline pitfall: “Thanks” pushes positive.
- Diagnostics: negation density high; entity masking keeps “1.2.4” neutral.
- Fix: boilerplate filter (polite preface), negation window on “doesn’t” ⇒ negative; confidence high (no sarcasm cues).

4.3 Product reviews (irony, hedged praise)
Text E: “Not bad at all; could be faster, but the battery is **killer**.”

- Signals: double positive (“not bad at all”, “killer”=positive sense), one modest negative.
- Baseline pitfall: “killer” treated as negative; “but” mis-aggregated.
- Diagnostics: negation window flips “bad”; domain sense list maps “killer”→positive; contrast weights the “could be faster” as small negative.
- Fix: sense disambiguation + contrast weighting ⇒ mildly positive with a speed complaint tag.

Text F: “Exactly what I needed… **after** I disabled the default settings.”

- Signals: praise constrained by prerequisite (setup pain).
- Baseline pitfall: reads positive overall.
- Diagnostics: sarcasm proxy low, but segment labels (Outcome: positive; Setup: negative).
- Fix: two-label output: “Positive outcome, negative setup UX”; send setup issue to onboarding team.

4.4 Workplace email (polite negativity, faint praise)
Text G: “Could we maybe revisit the timeline? The current dates look **ambitious**.”

- Signals: hedges + euphemism (“ambitious” = too tight).
- Baseline pitfall: “ambitious” seen as positive.
- Diagnostics: intensifier/hedge table active; euphemism list maps ambitious→schedule_risk in email genre.

- Fix: genre gate = Instruction/Request; convert to neutral sentiment with **risk flag** for PM dashboard.

Text H: "Appreciate the quick turn. I'm still seeing **regressions** in the auth flow."

- Signals: faint praise + hard negative entity.
- Baseline pitfall: averages to neutral.
- Diagnostics: segment peaks (praise then neg); entity "regressions/auth" dominates problem segment; entropy low.
- Fix: segment-aware fusion: overall negative for quality; preserve appreciative sentence; route to security/auth owners.

Ops patterns across cases

- Always **segment** when you see conjunctions ("but/though/after"), list markers, or paragraph breaks.
- Maintain **domain lexicons** (slang, euphemisms, event terms) and refresh monthly.
- Use **sarcasm proxy** + **negation density** as gates; if high and transformer entropy > 0.6, **abstain/route**.
- Log **disagreement** between EmbeddingBag and transformer; sample for weekly human review.
- Emit **two labels** when appropriate (e.g., Praise + Problem) instead of flattening to one score.

5) Mitigations Without Massive Training

5.1 Windowed negation rules; intensifier tables (no parser required)

- **Negators:** {not, no, never, n't, hardly, scarcely, barely, no_longer}.
- **Scope window:** flip the *next* eligible sentiment token within W=4 tokens or until punctuation {. ? ! ; :} or the conjunction *but*.
- **Intensifiers/attenuators:** apply a multiplier to the *next* eligible sentiment token within 2 tokens.
    - Boosters: {very:1.5, so:1.4, really:1.3, super:1.3, unbelievably:1.6}

    - Softeners: {slightly:0.7, somewhat:0.8, kind_of:0.85, a_bit:0.9}

- **Edge rules:** if both negation and intensifier hit the same token, apply negation first (flip), then multiply.
- **Minimal pseudocode:**
- for i,t in enumerate(tokens):
- if t in NEGATORS: mark window [i+1, i+W] until punct/but
- if t in BOOSTERS: tag next sentiment token with mult>1
- if t in SOFTENERS: tag next sentiment token with mult<1
- if t is sentiment_token:
- s = base_lexicon_score(t)
- if in_negation_window: s = -s; clear_window
- if has_multiplier_tag: s = s * mult; clear_tag
- acc += s
- net = acc / sqrt(L)

5.2 Tiny domain lexicons (per team or product)

- **Why tiny works:** Most errors come from a handful of local terms (slang, euphemisms, product jargon). A 50–200 word list per domain captures 80/20.
- **Process:**
    1. Sample 200–500 recent items; list top tokens by TF–IDF or attention attribution.
    2. Human-in-the-loop: mark tokens as {pos, neg, neutral, sense-dependent}.
    3. For sense-dependent terms (e.g., *killer*, *sick*), add **context triggers**: killer + battery => pos; killer + bug => neg.
    4. Version and date-stamp each lexicon; refresh monthly or when drift alarms fire.
- **Implementation tips:**
    - Keep a single JSON per stream: {"pos":[], "neg":[], "rules":[{"match":["killer","battery"],"score":+2}]}
    - Apply before model aggregation; unknowns default to 0.

5.3 Genre gating (Complaint/Praise/Irony/Instruction) before sentiment

- **Rationale:** Sentiment behaves differently across genres; gating reduces false positives.
- **Lightweight gate (no training required):**
    - **Complaint cues:** fail, crash, broken, refund, issue, cannot, doesn't, blocked, outage.
    - **Praise cues:** love, thanks, works, impressed, delight, awesome, smooth.
    - **Instruction cues:** please, could you, steps, repro, attach, follow, configure.
    - **Irony cues:** ellipses ..., scare quotes "great", contrastive *yeah right*, positive word + negative event.
- **Decision:** choose the **argmax** of cue counts (ties => Unknown).
- **Downstream rule:**
    - If *Instruction* or *Unknown*, cap absolute sentiment (e.g., clamp to [-0.7, +0.7]) and emphasize **routing** over scoring.
    - If *Irony*, apply sarcasm down-weight (see 5.4) and raise abstention threshold.
    - If *Complaint* and *Praise* both fire, **segment** the text and score segments separately.

5.4 Sarcasm heuristics (contrastive cues, punctuation, emoji sets)

- **Heuristic features (0..1 proxy):**
    - Positive adjective + negative event within 5 tokens (e.g., *amazing* + *crash/refund/delay*) => +0.5
    - Ellipses ... or scare quotes around sentiment token => +0.2
    - Contrastive conjunction after praise: *but/though/however* within 5 tokens => +0.2
    - Ironic emojis: 🙃 😬 😒 😑 combined with positive words => +0.2 (cap overall at 1.0)
- **Use:**
    - Plot as a line for analysts.
    - Down-weight net sentiment: net_adj = net * (1 - 0.5*sarcasm_proxy)

    - If sarcasm_proxy >= 0.6 and model entropy is high (see 5.5), **abstain** and route to review.

5.5 Calibration and abstention ("low confidence" routing)

- **Temperature scaling (transformer head):**
    - Fit scalar T on a small validation set to minimize NLL.
    - Inference uses softmax(logits / T). Store T per stream (social, support, email).
- **Entropy bands:**
    - Compute normalized entropy H_norm = H(p) / log(K) in [0,1].
    - Thresholds:
        - H_norm <= 0.3: **high confidence** → allow automation (e.g., triage tag).
        - 0.3 < H_norm <= 0.6: **medium** → log, show in dashboard.
        - H_norm > 0.6: **abstain/route** to human.
- **Dual-check agreement:**
    - Compare sign of (EmbeddingBag net) vs (transformer p_pos - p_neg).
    - If **disagree** and H_norm > 0.5, flag for review, store in weekly error notebook.
- **Drift sentinels:**
    - Track running mean/std of negation density, sarcasm proxy, token length.
    - If any exceed 3*sigma relative to a 30-day baseline, raise a **drift alert**; freeze auto-actions until a sample is audited.

5.6 Boilerplate & segmentation (cheap, high impact)

- **Boilerplate filters:** strip greetings, signatures, legal footers, and ticket templates before scoring.
- **Segmentation:** split on bullets, hard breaks, or conjunctions (*but/though*). Score segments, then **aggregate** with:
    - Complaint-first: min(segment_nets) for support routing, or
    - Weighted fusion: weight by presence of problem entities (export, refund, auth).

- **Output dual labels when warranted:** e.g., Praise + Problem, Outcome + Setup.

5.7 Defaults you can copy/paste today

- Negation window W=4; intensifier window=2.
- Sarcasm down-weight factor=0.5; abstain if sarcasm_proxy>=0.6 and H_norm>0.5.
- Clamp sentiment in Instruction/Unknown genres to [-0.7,+0.7].
- Tiny lexicon size per stream: 100–200 entries; refresh monthly.
- Weekly QA: review 50 **disagreements** (baseline vs transformer) and 50 **abstentions**; update lexicons/rules accordingly.

These mitigations keep everything **local, fast, and interpretable**: no giant retrains, just small tables and rules that measurably reduce the most common, costly mistakes.

6) Dashboards for Analysts

Multi-line plots, delta views, drift monitors, error notebooks

6.1 Multi-line plots (always-on status panel)
Purpose: show *what moved* and *why* at a glance.
Contents per stream (social, support, reviews, email), time-bucketed (e.g., hourly, daily):

- Lines: pos, neg, net, negation_density, sarcasm_proxy, avg_len (secondary axis if needed), optional transformer p_pos (mean).
- Tooltips: sample size n, min/median/max of net, top drivers (top 3 tokens by absolute contribution).
- Controls: time window picker, stream filter, translation/domain toggle, “normalize by z-score” switch.
  Export: scores_by_time.csv with fields
  ts, stream, n, pos_mean, neg_mean, net_mean, neg_density_mean, sarcasm_mean, len_mean, ppos_mean, pneg_mean, pneu_mean.

6.2 Delta views (responsiveness & anomaly triage)
Purpose: locate *where* change happened.

- Per item index or per time bucket, plot absolute first differences:
  |Delta(pos)|, |Delta(neg)|, |Delta(net)|, |Delta(negation_density)|, |Delta(sarcasm_proxy)|.

- Sorting modes: largest delta first; delta per segment (for tickets with bullets/paragraphs).
- Drill-down: click a spike → open the item with token attributions and applied rules (negation window hits, intensifier multipliers, sarcasm cues).
  Export: deltas.csv
  id, ts, stream, d_pos, d_neg, d_net, d_neg_density, d_sarcasm, len, genre_gate, abstained(bool).

6.3 Drift monitors (early-warning sentinels)
Purpose: detect distribution shift before it wrecks precision/recall.
Metrics tracked as rolling z-scores against a 30-day baseline:

- negation_density, sarcasm_proxy, token_length, OOV_rate, emoji_rate, hashtag_rate, entity_mask_rate.
- Optional label-side sentinels (if you have light labels): complaint_rate, praise_rate, abstention_rate.
  Alerts: trigger when any metric's rolling mean exceeds 3*sigma for 3 consecutive buckets or when KL divergence vs baseline exceeds a threshold.
  Pseudocode (copy-safe):

```
for metric in METRICS:
  z = (mean_recent(metric) - mean_baseline(metric)) / std_baseline(metric)
  if z > 3.0 for 3+ consecutive buckets: raise_alert(metric)
if KL(p_recent || p_baseline) > kl_thresh: raise_alert("distribution")
```

UI: small sparklines with traffic-light badges; clicking opens a "what changed" panel (top token shifts, new slang, entity spikes).
Export: drift_log.csv
ts, stream, metric, z_score, window, baseline_ref, alert(bool).

6.4 Error notebooks (human-in-the-loop fixes)
Purpose: convert errors into rules/lexicon updates.
Daily auto-populate three tabs:

- **Disagreements:** EmbeddingBag sign vs transformer sign differ AND entropy > 0.5.
- **Abstentions:** Items with entropy > 0.6 or sarcasm_proxy >= 0.6.
- **Misroutes (if labeled):** Wrong queue or wrong sentiment.
  Each row shows: raw text, tokens, applied negation windows, intensifier hits, sarcasm

cues, entity masking, segment scores, final decision, confidence.
Analyst actions (single-click):

- Add lexicon entry (+/-) or context rule (e.g., "killer"+"battery" => +2).
- Adjust negation window W or intensifier weights for this stream.
- Mark genre override (Complaint/Praise/Irony/Instruction).
- Accept abstention and assign to reviewer.
  Changes are versioned: lexicon_v, rules_v, calib_v with timestamps and author.
  Exports:
- error_notebook_disagreements.csv
- error_notebook_abstentions.csv
- patchset.json (atomic bundle of lexicon/rule/calibration updates).

6.5 Item drill-down (explainability card)
For any item, render a compact card:

- Header: stream, ts, id, genre_gate, abstained?.
- Scores: pos, neg, net, len_norm, sarcasm_proxy, transformer probs, entropy.
- Rationale:
  - Top contributing tokens with weights; negation window spans [i..j]; intensifier multipliers; sarcasm feature hits.
  - Segment table (if any): segment_type, segment_text, segment_net.
- Suggested fixes: rules that would have flipped/tempered the outcome (what-if simulation).

6.6 KPIs & guardrails panel

- Precision@high-confidence, recall on complaint slices, abstention_rate, disagreement_rate, ECE (calibration).
- SLA badges: “auto-route allowed when precision@HC >= target and drift alerts == 0”.
- Review cadence widget: % of weekly quota completed; oldest pending abstentions.

6.7 Minimal file contract (so dashboards are portable)

- scores.csv (per item), scores_by_time.csv (aggregates), deltas.csv, drift_log.csv, error_notebook_*.csv, patchset.json, calib.json.
- Field names stable; no proprietary binary blobs; everything diff-able and auditable.

6.8 Workflow in practice

1. Morning: check drift badges and KPIs; if red, pause auto-actions.
2. Work the **Disagreements** and **Abstentions** queues (top deltas first).
3. Apply a **patchset** (tiny lexicon/rule tweaks); re-compute a 24-hour backfill; verify KPIs.
4. Weekly: export an **error report** (top 20 recurring issues), refresh domain lexicons, recalibrate temperature if needed.

These dashboards keep the system legible: analysts see *movement* (multi-line), *surprises* (deltas), *shifts* (drift), and *fix paths* (notebooks), turning sentiment from a black box into an instrument panel you can actually fly.

7) Evaluation & Monitoring

Balanced test slices; human review loops; regression alarms

7.1 Goals & principles

- **Measure what matters in the wild.** Optimize for correct routing/action, not leaderboard accuracy.
- **Balance rigor with cost.** Use small but representative **slices** and continuous sampling.
- **Prefer calibrated, abstaining systems.** A safe "I'm not sure" beats a confident wrong.

7.2 Test set construction (balanced slices)

Create a compact, evergreen "golden" corpus (e.g., 1–2k items) stratified by domain and failure risks:

- **By genre:** Complaint, Praise, Irony/Sarcasm, Instruction/Neutral.
- **By cues:** High negation, high intensifiers, emoji-heavy, hashtag-heavy, code-switching/AAVE, boilerplate-heavy.
- **By entities:** Brand/product names, technical words with neutral senses (crash, bug, killer feature).

- **By length/structure:** Short posts, long emails, bullet lists, mixed praise/problem tickets. Target ~100–150 items per slice; refresh 10–20% monthly to avoid overfitting.

7.3 Labeling protocol & quality

- **Guidelines:** Define sentiment of **author stance** (not quoted speech). Document rules for negation scope, sarcasm, euphemism.
- **Double annotation** on 30–50% of items; **adjudication** on disagreements.
- **Reliability:** Track **Cohen's κ** per slice (aim $\kappa \geq 0.70$). Revisit guidelines where $\kappa < 0.6$.
- **Multi-label** allowed (e.g., Praise + Problem). Record **rationale notes** for edge cases.

7.4 Core metrics (per slice + macro)

- **Classification:** F1 (macro), precision/recall per class; **MCC** for robustness on imbalanced data.
- **Calibration: ECE** (expected calibration error), **Brier score**.
- **Uncertainty: Abstention rate** and **selective risk** (error rate on auto-acted items).
- **Ranking (triage):** AUROC/PR-AUC for "Complaint vs other"; **Kendall τ** for priority ranking where applicable.
- **Fairness:** ΔF1 and ΔECE across dialect or demographic proxies (where ethically permissible).

7.5 Acceptance thresholds (illustrative defaults)

- Macro F1 ≥ 0.80 on **Complaint** slices; ≥ 0.75 overall.
- ECE ≤ 0.07; Brier ≤ 0.18.
- Selective risk ≤ 2.5% at **high-confidence band** ($H_{norm} \leq 0.3$).
- Abstention 10–25% (tunable by ops load).
- Disagreement rate (baseline vs transformer) ≤ 12% and trending down week-over-week. Tune these by application severity (e.g., support routing vs public moderation).

7.6 Human review loops (continuous improvement)

- **Daily queue:** n=50 items prioritized by (a) high entropy, (b) large deltas, (c) model disagreement.

- **Weekly clinic:** Review top 100 recurring errors; update tiny lexicons, negation/intensifier tables, sarcasm cues.
- **Patchsets:** Bundle rule/lexicon updates with version stamps; re-score last 24–72h in **shadow** to verify KPIs before enabling.

7.7 Regression testing & alarms

- **Frozen golden set:** Run on every change; fail build if any KPI drops > δ (e.g., macro F1 −2 pts, ECE +0.02, selective risk +0.5 pt).
- **Time-backtest:** Recompute last 30 days; compare drift of means/variances and error pockets.
- **Canary deploy:** 10% traffic with new patch; monitor **regression alarms** on:
  - complaint recall ↓ > 3 pts,
  - abstention ↑ > 10 pts,
  - drift z-score > 3 on negation_density/sarcasm_proxy/token_length,
  - elevated disagreement rate.

7.8 Drift detection & monitoring

- **Covariate drift:** KL or PSI on token/emoji/hashtag/length distributions; rolling **z-scores** vs 30-day baseline.
- **Label drift (if labels exist):** Prior shift in slice proportions (Complaint/Praise/Irony).
- **Concept drift:** Rise in error residuals for stable slices; sudden metric divergence while inputs look similar.
- **Response:** Lock auto-actions, trigger sampling + analyst review; update domain lexicons and calibration temperatures, then canary.

7.9 Calibration management

- **Per-stream temperature scaling** (social/support/reviews/email); re-fit monthly or when ECE > 0.1.
- **Entropy bands:** Review thresholds quarterly; align with staffing capacity.
- **Agreement gate:** If transformer sign ≠ EmbeddingBag sign and H_norm > 0.5 → auto-abstain + log to error notebook.

7.10 Reporting & governance

- **Scorecards (weekly):** Macro F1, per-slice F1, ECE, selective risk, abstention, disagreement, drift alerts, fairness deltas.
- **Error taxonomy:** Track counts by cause (negation miss, sarcasm miss, slang miss, quote stance, boilerplate bleed, entity leakage).
- **Red teaming:** Quarterly probes for adversarial phrasing (e.g., polite negativity, ironic praise).
- **Privacy & audit:** Retain only minimal text needed for diagnostics; log all rule/lexicon/calibration changes with author/time.

Checklist (ship gate)

- Golden set KPIs ≥ thresholds
- Calibration within ECE target; entropy bands configured
- Drift monitors green for 7 days
- Human review queue staffed; error notebook workflow active
- Canary passed (no regression alarms)
- Patchset versioned and reproducibility artifacts exported (scores.csv, deltas.csv, drift_log.csv, calib.json)

With these practices, you don't just "have a number"—you have a **living measurement system** that detects when language shifts, explains why scores moved, and gives clean levers for safe, incremental improvement.

8) Ethics & Risk

Dialect fairness, workplace impact, moderation sensitivity

8.1 Dialect fairness (code-switching, AAVE, sociolects)

- **Risk:** Models conflate dialectal form with negativity or toxicity, over-flagging AAVE or non-native English while under-flagging "polite" hostility in prestige dialects.
- **Harms:** Disparate false positives in hiring screens, customer moderation, and support triage; chilled speech in marginalized communities.

- **Mitigations:**
    - **Slice-aware evaluation:** Always report metrics by dialect proxies (curated AAVE sets, non-native error patterns, code-switch examples). Track ΔF1 and ΔECE across slices.
    - **Genre/dialect gate before sentiment:** Route dialectal text through adjusted priors (reduced weight on grammar cues; boosted reliance on content terms).
    - **Counterfactual auditing:** Paraphrase items into multiple dialects; require decision invariance where semantics are preserved.
    - **Human review on flagged cohorts:** Create a standing QA lane for dialectal content; prioritize false-positive reduction.
    - **Data hygiene:** Avoid training on scraped "toxicity" datasets that embed community stereotypes without context; document sources.

8.2 Workplace impact (email, tickets, performance signals)

- **Risk:** Sentiment scores bleed into performance dashboards, quietly shaping reviews, promotion narratives, or vendor penalties. "Polite negativity" in some teams becomes noise elsewhere; uneven lexicons punish specific roles (e.g., QA reporting "fail," "bug," "regression").
- **Harms:** Incentivizes language theater over candor; penalizes roles tasked with surfacing risk; skews leadership perception.
- **Mitigations:**
    - **Purpose binding:** Explicitly prohibit using raw sentiment for individual performance assessment. Use it for **routing and escalation**, not HR judgments.
    - **Role-aware normalization:** For QA/SRE streams, downweight problem tokens; score by **resolution trajectory** (problem → fix) rather than raw polarity.
    - **Dual labels > single score:** Preserve "Praise + Problem" structure; never collapse into one number for personnel decisions.
    - **Transparency & appeal:** Visible audit trail (item view with rules, windows, lexicons). Employees can contest or annotate misreads.
    - **Governance cadence:** Ethics review with worker reps; quarterly red-team focusing on internal communications.

8.3 Moderation sensitivity (public forums, brand channels)

- **Risk:** Over-removal of legitimate criticism; under-removal of harm veiled in politeness or irony. Sarcasm and reclaimed slurs complicate automatic flags.
- **Harms:** Suppresses dissent; erodes trust; enables reputational manipulation.
- **Mitigations:**
    - **Abstention as policy:** High-uncertainty or high-sarcasm items must route to humans. Set non-negotiable thresholds (e.g., entropy > 0.6 or sarcasm_proxy ≥ 0.6).
    - **Context windows:** Evaluate posts with the immediate reply chain or pericope; single-utterance scoring is advisory only.
    - **Appeals and restoration:** Fast appeal channel; restore content when heuristics—not intent—drove the decision.
    - **Separation of concerns:** Distinguish **harm categories** (harassment, threat) from **sentiment**. Do not equate negativity with policy violation.
    - **Localized lexicons:** Region/language-specific slang and reclaimed terms with documented community usage; treat them as **ambiguous** by default.

8.4 Bias, privacy, and explainability

- **Bias monitoring:** Publish fairness deltas (ΔF1, ΔECE) on key slices; set regression alarms when gaps widen.
- **Privacy by design:** Minimize retention; redact PII; keep only short text spans and derived features needed for diagnostics.
- **Explainable outputs:** Every automated action must have an item card: token attributions, negation window spans, intensifier multipliers, sarcasm cues, final confidence.
- **Provenance & patch control:** Version and sign lexicons/rules; maintain a change log with author and rationale to prevent silent policy drift.

8.5 Guardrails & decision policy

- **No single score as a gate.** Require agreement between transformer and baseline or else abstain.
- **Confidence bands tied to action.** Only "high confidence" items trigger automation; the rest are advisory or routed.
- **Drift kills automation.** Any 3-sigma drift alert in negation density, sarcasm proxy, or token length pauses auto-actions for that stream pending audit.
- **Human-in-the-loop as principle.** Document staffing for review queues before deploying auto-moderation or auto-routing.

8.6 Documentation (model cards & process)

- **Model card:** Training sources, known limitations (negation, sarcasm, dialect), supported domains, calibration status, abstention policy.
- **Data card:** Collection methods, consent/terms, demographic and dialect coverage, known gaps.
- **Ops SOPs:** Escalation playbook, appeal turnaround SLAs, weekly error notebook workflow.

Bottom line: Treat sentiment as **instrumentation**, not judgment. Protect people first, especially dialect communities and employees whose job is to report problems. Prefer calibrated, abstaining systems with transparent rationales over brittle, high-throughput automation.

9) Limitations & Future Work

Beyond heuristics; weak supervision; small fine-tunes; multimodal cues

9.1 Limitations of the current stack (what we did on purpose)

- **Heuristic bias & ceiling effects.** Windowed negation, intensifier tables, sarcasm proxies, and tiny lexicons are fast and interpretable—but brittle. They plateau on out-of-domain phrasing, clever sarcasm, and long cross-clause dependencies.
- **Verse/post granularity.** Our unit is a single line/segment; many errors live in **discourse** (thread state, author intent across turns) that we only approximate via segment aggregation.

- **Transformer optionality without training.** The “drop-in head” is only as good as its unseen pretraining/fine-tune distribution; calibration helps, but true **concept drift** remains.
- **Limited semantics.** No coreference, no quote attribution beyond heuristics, no tense/aspect modeling (e.g., “used to fail” vs “fails”), and minimal sense disambiguation (e.g., killer bug vs killer battery).
- **Fairness scope.** We track dialect slices but don’t yet provide provable guarantees; mitigation remains process-based (review queues, slice metrics) rather than certified bounds.

9.2 Beyond heuristics: structural upgrades

- **Shallow dependency cues.** Replace fixed windows with 1-pass dependency hints (negation governors, scope via conjunction boundaries); keep it < O(n) and deployable.
- **Quote & stance module.** Lightweight detector for quoted spans + cue words (“he said”, “according to”), with stance rules (reported ≠ endorsed).
- **Contrastive sense tables.** Move from word lists to **(term, context)** pairs with priorities, e.g., (“crash”, software) → neutral; (“crash”, car) → negative.
- **Segment graph fusion.** Model a message as a small graph (segments as nodes, contrast/causal edges) and aggregate sentiment with edge-aware weights (e.g., “but” edges down-weight earlier praise).

9.3 Weak supervision (scale labels without scale cost)

- **Programmatic labeling.** Convert our rules into **labeling functions** (LFs) and combine with a label model (e.g., Snorkel-style) to produce probabilistic labels for thousands of items.
- **Heuristic ensemble diversity.** Author multiple LFs per phenomenon (negation, irony, domain sense) to reduce single-rule bias; learn LF accuracies and correlations.
- **Teacher–student bootstraps.** Use the LF label model (teacher) to fine-tune a small student classifier (BiLSTM/Transformer-mini) for the target stream; retain explainability by logging LF votes.

9.4 Small, targeted fine-tunes (hours, not weeks)

- **Domain-adaptive pretraining (DAPT-lite).** Continue pretraining a small transformer (e.g., 50–150M params) for a few thousand steps on **in-domain unlabeled** text (support emails, reviews).

- **Task-adaptive fine-tuning (TAPT).** Fine-tune on a few hundred **human-audited** items plus weak labels; freeze embeddings, train upper layers; early-stop via slice F1/ECE.
- **Multi-head outputs.** Add heads for genre (Complaint/Praise/Irony/Instruction) and uncertainty, sharing a common encoder; improves calibration and routing.

9.5 Multimodal cues (where text alone will keep missing intent)

- **Emojis & typography as first-class features.** Treat emoji sequences, ALL CAPS bursts, and punctuation patterns as learned embeddings, not post-hoc heuristics.
- **Thread & metadata.** Include reply depth, time gaps, sender role (QA vs PM), platform (social vs ticketing), and known entities (version ids) as side inputs.
- **Audio/visual for select domains.** In support or social, attach lightweight audio sentiment (prosody) or image cues (e.g., screenshot with error dialog detection). Keep these optional and privacy-gated.

9.6 Calibration & abstention 2.0

- **Per-slice temperature vectors.** Instead of a single T per stream, learn T_slice for key slices (negation-heavy, emoji-heavy, dialect), chosen by a gate at inference.
- **Selective prediction with coverage guarantees.** Train for **risk-controlled** prediction (set-valued outputs) to bound error on auto-actions; maximize coverage under a target risk.
- **Counterfactual calibration.** Evaluate on counterfactuals (mask entities, flip dialect) to ensure confidence doesn't spike spuriously.

9.7 Fairness & governance road map

- **Targeted augmentation.** Generate dialectal and paraphrase variants for every high-impact template; enforce decision invariance in training and review.
- **Slice contracts.** Publish target ΔF1 and ΔECE caps across slices; wire regression alarms to deployment gates.
- **Data minimization & privacy.** Move toward feature-only storage (attributions, counts, spans), with ephemeral raw text; audit trail for every automated action.

9.8 Evaluation at scale (closing the loop)

- **Living golden sets.** Rotate 10–20% monthly; tag reasons-for-error; maintain a long-tail bucket (rare slang, new product terms).

- **Causal probes.** When metrics move, run ablations (remove negation rules, drop sarcasm cues) to attribute gains/regressions.
- **Shadow deployments.** Before enabling new rules or fine-tunes, score live traffic in shadow, compare selective risk and abstention, and only then flip.

9.9 What "good" looks like in the next iteration

- +5–10 pts macro F1 on **Complaint/Irony** slices with **no increase** in selective risk at high confidence.
- ECE ≤ 0.05 with per-slice T; abstention stable at 10–20% but **more informative** (higher error reduction per abstention).
- Documented fairness deltas within agreed caps; zero red drift alerts for 30 days post-deploy.
- Analyst time drops (fewer disagreements, faster patchsets) because dashboards point straight to actionable tokens/spans.

Bottom line: We stay local, fast, and interpretable—but graduate from static heuristics to **weakly supervised, lightly fine-tuned, slice-calibrated** models, enriched with minimal multimodal/context features. That path buys real-world robustness without the cost—or opacity—of heavyweight retrains.

10) Reproducibility Appendix

Minimal PyTorch scripts, lexicon seeds, negation rules, calibration tips

---

A. Minimal PyTorch baseline (single file, no downloads)

Save as sentiment_baseline.py and run: python sentiment_baseline.py input.txt

```
# sentiment_baseline.py
# Usage: python sentiment_baseline.py input.txt
# input.txt: one document per line (tweet, verse, ticket, email paragraph)

import sys, re, math, json
import torch
import pandas as pd
import matplotlib.pyplot as plt

assert len(sys.argv) == 2, "Usage: python sentiment_baseline.py input.txt"
path = sys.argv[1]
docs = [ln.strip() for ln in open(path, "r", encoding="utf-8") if ln.strip()]

def toks(s):
    return [t for t in re.split(r"[^A-Za-z]+", s.lower()) if t]

# --- tiny, editable lexicons (seed lists; edit freely) ---
POS = {
    "love":3, "kind":2, "truth":2, "awesome":2, "great":2, "smooth":1,
    "hope":2, "faith":2, "perfect":2, "rejoice":2, "amazing":2
}
```

```
NEG = {
   "fail":-2, "failing":-2, "failed":-2, "bug":-2, "crash":-2, "delay":-1,
   "nothing":-2, "iniquity":-2, "evil":-2, "darkly":-1, "regression":-2
}

NEGATORS = {"not","no","never","n't","hardly","scarcely","barely","no_longer"}
INTENSIFY = {"very":1.5,"so":1.4,"really":1.3,"super":1.3,"unbelievably":1.6}
SOFTEN   = {"slightly":0.7,"somewhat":0.8,"kind_of":0.85,"a_bit":0.9}

def windowed_score(tokens, W=4):
   acc = 0.0
   flip_until = -1
   mult_next = 1.0
   for i,t in enumerate(tokens):
      if t in NEGATORS:
         flip_until = i + W
         mult_next = 1.0
         continue
      if t in INTENSIFY:
         mult_next = max(mult_next, INTENSIFY[t]); continue
      if t in SOFTEN:
         mult_next = min(mult_next, SOFTEN[t]); continue
      base = 0.0
      if t in POS: base += POS[t]
      if t in NEG: base += NEG[t]
      if base != 0.0:
```

```
            if i <= flip_until: base = -base; flip_until = -1
            base *= mult_next
            mult_next = 1.0
            acc += base
        # reset scope at punctuation boundaries (simulate clause end)
        # (only works if you pass punctuation as tokens; we don't here)
    L = max(1, len(tokens))
    return acc / math.sqrt(L), L

rows = []
for idx,doc in enumerate(docs, start=1):
    tokens = toks(doc)
    net, L = windowed_score(tokens)
    # naive pos/neg components (without negation) for diagnostics
    pos = sum(POS.get(t,0) for t in tokens)/math.sqrt(max(1,L))
    neg = sum(NEG.get(t,0) for t in tokens)/math.sqrt(max(1,L))
    neg_density = sum(1 for t in tokens if t in NEGATORS)/max(1,L)

rows.append({"id":idx,"len":L,"pos":pos,"neg":neg,"net":net,"negation_density":neg_density,"t
ext":doc})

df = pd.DataFrame(rows)
df.to_csv("scores.csv", index=False, encoding="utf-8")

# one chart, multiple lines (no explicit colors)
plt.figure()
plt.plot(df["id"], df["pos"], marker="o", label="pos")
```

```
plt.plot(df["id"], df["neg"], marker="o", label="neg")
plt.plot(df["id"], df["net"], marker="o", label="net")
plt.plot(df["id"], df["negation_density"], marker="o", label="negation_density")
plt.title("Multi-line diagnostics (baseline)")
plt.xlabel("item id"); plt.ylabel("score / density")
plt.grid(True, linestyle="--", linewidth=0.5); plt.legend()
plt.savefig("diagnostics.png", dpi=150, bbox_inches="tight")
print("Wrote: scores.csv, diagnostics.png")
```

Notes

- Completely local; edit POS/NEG seeds as you like.
- windowed_score implements negation scope and intensifiers with constant-time passes.
- Normalization sum/sqrt(L) reduces length bias.

B. Negation rules (drop-in table & behavior)

- **Scope window:** default W=4 tokens after a negator; stop early at punctuation or coordinating “but/though/however” (if you tokenize punctuation).
- **Priority:** negate first, then apply intensifier/softener multiplier.
- **One-flip budget:** flip only the next eligible sentiment token; do not invert the entire tail (reduces false flips).

Editable table (JSON example for reuse across scripts):

```
{
  "negators": ["not","no","never","n't","hardly","scarcely","barely","no_longer"],
  "boosters": {"very":1.5,"so":1.4,"really":1.3,"super":1.3,"unbelievably":1.6},
  "softeners": {"slightly":0.7,"somewhat":0.8,"kind_of":0.85,"a_bit":0.9},
  "scope_window": 4,
  "stop_tokens": [".","!","?",";":":","but","though","however"]
}
```

C. Tiny, domain-specific lexicon seeds (start kits)

Copy/paste these into your POS/NEG dicts per stream, then evolve weekly.

Social (slang core):

- POS: sick, fire, legit, dope, based, clutch, wholesome, win, glow-up
- NEG: cringe, flop, L, cope, mald, ratioed, downbad, bugged, lag, queue

Support/Tickets:

- POS: resolved, fixed, stable, green, smooth, restored, mitigated
- NEG: fail, failing, crashed, timeout, outage, blocked, regression, corrupted, memory-leak

Reviews:

- POS: snappy, responsive, intuitive, rock-solid, killer (battery/sound), buttery
- NEG: sluggish, janky, brittle, flaky, bloat, overheats, throttling

Email/Workplace:

- POS: appreciate, thanks, helpful, aligned, unblocked, clarified
- NEG: regressions, slipping, risky, overcommitted, ambiguous, bottleneck

Sense rules (examples):

- ("killer","battery|sound|feature") => +2
- ("crash","software|app|export") => 0 (neutral tech sense)
- ("crash","car|market|server-farm") => -2

Represent as JSON rules:

```
[
 {"match":["killer"],"context_any":["battery","sound","feature"],"score":2},
 {"match":["crash"], "context_any":["app","export","software"], "score":0},
 {"match":["crash"], "context_any":["car","market"], "score":-2}
]
```

D. Sarcasm proxy (cheap, explainable)

Add to your pipeline; plot it as a 0..1 line and use it to down-weight net scores.

IRONIC_EMOJIS = {"🙃","😬","😒","😑"}

NEG_EVENTS = {"crash","refund","delay","outage","bug","fail","failing","broken"}

```
def sarcasm_proxy(tokens, text):
    pos_hits = any(t in {"great","amazing","perfect","awesome"} for t in tokens)
    neg_hits = any(t in NEG_EVENTS for t in tokens)
    ellipses  = ("..." in text)
    scare_q   = bool(re.search(r'\"(great|amazing|perfect|awesome)\"', text.lower()))
    contrast  = any(w in tokens for w in ["but","though","however","yeah","right"])
    emojis    = any(e in text for e in IRONIC_EMOJIS)
    score = 0.0
    if pos_hits and neg_hits: score += 0.5
    if ellipses or scare_q:   score += 0.2
    if contrast:              score += 0.2
    if emojis:                score += 0.2
    return min(1.0, score)
```

Down-weighting:

net_adj = net * (1 - 0.5 * sarcasm_proxy(tokens, doc))

E. Transformer head (optional, local only) + temperature scaling

If you have a **local** fine-tuned model folder (no internet), add calibrated probabilities.

```
# transformer_head.py (snippet)
from transformers import AutoTokenizer, AutoModelForSequenceClassification
import torch, numpy as np
```

```
def load_head(local_dir):
    tok = AutoTokenizer.from_pretrained(local_dir, local_files_only=True)
    mdl = AutoModelForSequenceClassification.from_pretrained(local_dir, local_files_only=True)
    mdl.eval(); return tok, mdl

def softmax(z):
    z = z - z.max(); e = np.exp(z); return e / e.sum()

def predict_probs(texts, tok, mdl, T=1.0):
    probs = []
    with torch.no_grad():
        for x in texts:
            inputs = tok(x, return_tensors="pt", truncation=True, max_length=256)
            logits = mdl(**inputs).logits[0].cpu().numpy()
            p = softmax(logits / max(1e-6, T))
            probs.append(p)
    return np.vstack(probs)  # shape [N,K]
```

Temperature scaling (fit once on a small labeled set):

```
# calib_temperature.py (snippet)
# y_true in {0..K-1}, logits: np.ndarray [N,K]
import numpy as np
from scipy.optimize import minimize

def nll_with_T(T, logits, y_true):
    T = max(1e-6, T[0])
```

```
    z = logits / T
    z = z - z.max(axis=1, keepdims=True)
    p = np.exp(z); p = p / p.sum(axis=1, keepdims=True)
    nll = -np.log(p[np.arange(len(y_true)), y_true] + 1e-12).mean()
    return nll

def fit_temperature(logits, y_true, T0=1.0):
    res = minimize(nll_with_T, x0=[T0], args=(logits, y_true), method="L-BFGS-B", bounds=[(1e-3, 100.0)])
    return float(res.x[0])
```

Store T per stream (e.g., calib.json):

```
{"social":{"T":1.31},"support":{"T":1.12},"reviews":{"T":1.22},"email":{"T":1.08}}
```

F. File schema (portable, dashboard-ready)

- scores.csv (per item):
  id, ts?, stream?, pos, neg, net, negation_density, sarcasm_proxy, len, p_pos?, p_neu?, p_neg?, entropy?
- diagnostics.png (one chart, multiple lines)
- deltas.csv (optional):
  id, d_pos, d_neg, d_net, d_neg_density, d_sarcasm
- calib.json: temperatures, thresholds, lexicon/rule versions
- patchset.json: proposed lexicon/rule updates with author/time

Entropy (for abstention bands), if you have p:

```
import numpy as np
def entropy(p):
    p = np.clip(p, 1e-12, 1.0); return float(-(p*np.log(p)).sum())
def entropy_norm(p):
```

H = entropy(p); return H / math.log(len(p))

G. Quick checklist (to reproduce our plots)

1. Put texts (one per line) in input.txt.
2. Run python sentiment_baseline.py input.txt → scores.csv, diagnostics.png.
3. (Optional) Add sarcasm_proxy and re-plot net vs net_adj.
4. (Optional) Add a local transformer head; fit T on a small validation set; save calib.json; re-score and plot entropy bands.
5. Version your seeds/rules: lexicon_v, rules_v, calib_v in calib.json.

H. Practical calibration tips

- **Small, balanced validation (n=200–500)** with clear labels across slices (Complaint, Praise, Irony, Instruction).
- **Optimize NLL** for temperature; verify **ECE** and **Brier** before/after.
- **Per-stream T** beats one global T (drift differs by stream).
- **Abstain thresholds**: start at entropy_norm > 0.6; tune to staffing capacity.
- **Agreement gate**: if transformer sign ≠ baseline sign and entropy_norm > 0.5 → abstain and log.

References

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review, 78*(1), 1–3. American Meteorological Society Journals

Councill, I. G., McDonald, R., & Velikovich, L. (2010). What's great and what's not: Learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in NLP* (pp. 51–59). Uppsala: University of Antwerp. ACL Anthology

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence, 2*(11), 665–673. Nature

Geifman, Y., & El-Yaniv, R. (2017). Selective classification for deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*. (Preprint). arXiv

Geifman, Y., Uziel, G., & El-Yaniv, R. (2019). SelectiveNet: A deep neural network with an integrated reject option. *ICML*. (Preprint). arXiv

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys, 46*(4), Article 44. netman.aiops.org

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. *ICML, PMLR 70*, 1321–1330. (Preprint and proceedings versions). Proceedings of Machine Learning Research+1

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., & Smith, N. A. (2020). Don't Stop Pretraining: Adapt language models to domains and tasks. *ACL*. (Preprint and ACL versions). arXiv+1

Hutto, C. J., & Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. *ICWSM*. AAAI Open Access Journal+1

Joshi, A., Bhattacharyya, P., & Carman, M. J. (2017). Automatic sarcasm detection: A survey. *ACM Computing Surveys, 50*(5), Article 73. (Preprint). ACM Digital Library+2CSE IIT Bombay+2

Kralj Novak, P., Smailović, J., Sluban, B., & Mozetič, I. (2015). Sentiment of emojis. *PLOS ONE, 10*(12), e0144296. (Preprint and journal versions). arXiv+1

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool. UIC Computer Science

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., … Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. (Preprint). arXiv+1

Paszke, A., Gross, S., Massa, F., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *NeurIPS*. (Preprint). NeurIPS Papers+1

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., & Ré, C. (2017). Snorkel: Rapid training data creation with weak supervision. *VLDB*. (Preprint). arXiv+1

Ratner, A., Bach, S. H., & Ré, C. (2016). Data programming: Creating large training sets, quickly. *NeurIPS*. (Preprint). arXiv

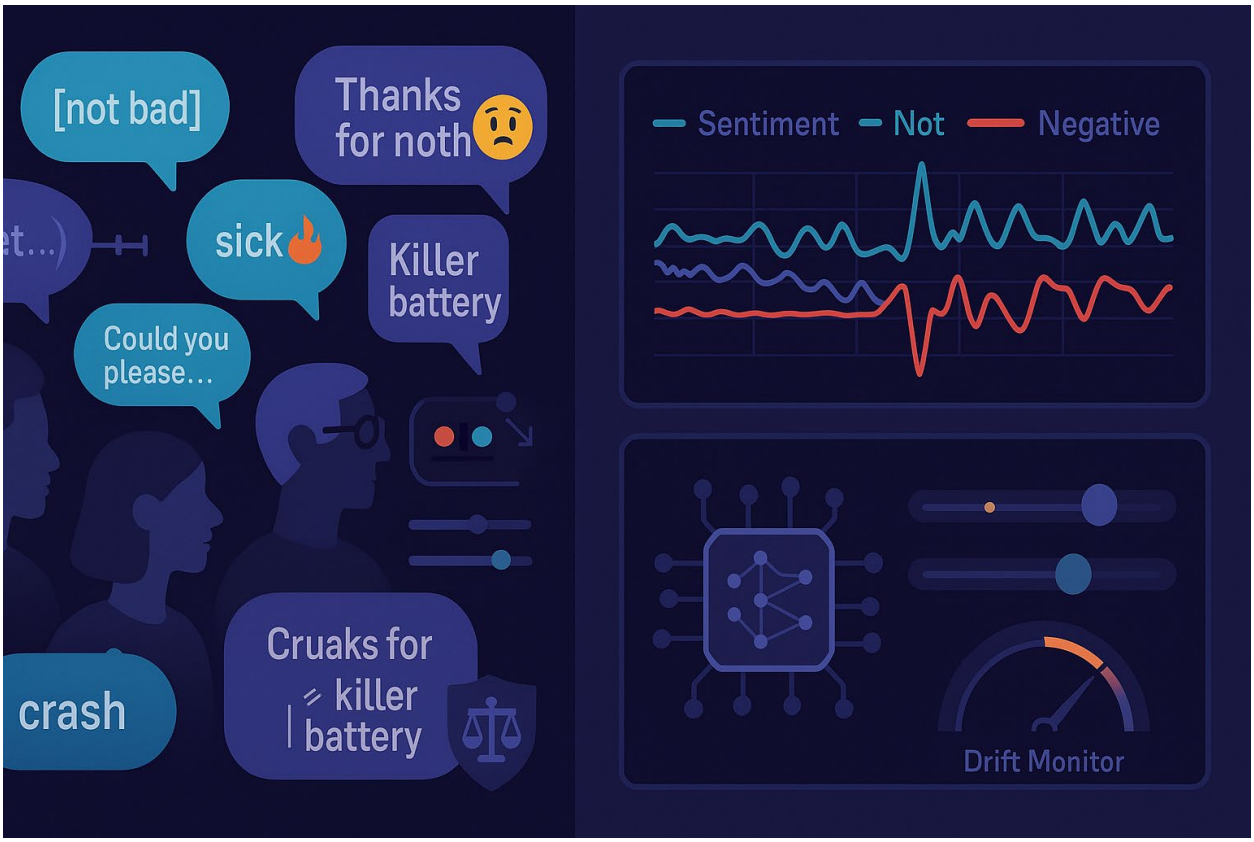
Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N., & Huang, R. (2013). Sarcasm as contrast between a positive sentiment and negative situation. *EMNLP* (pp. 704–714). (ACL and preprint versions). ACL Anthology+1

Sap, M., Card, D., Gabriel, S., Choi, Y., & Smith, N. A. (2019). The risk of racial bias in hate speech detection. *ACL*. ACL Anthology+1

Solorio, T., Blair, E., Maharjan, S., Bethard, S., Diab, M., Ghoneim, M., … Fung, P. (2014). Overview for the First Shared Task on Language Identification in Code-Switched Data. In *Proceedings of the First Workshop on Computational Approaches to Code Switching* (pp. 62–72). ACL Anthology

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP* (pp. 347–354). (ACL and preprint versions). ACL Anthology+1

Blodgett, S. L., Barocas, S., Daumé III, H., & Wallach, H. (2020). Language (technology) is power: A critical survey of “bias” in NLP. *ACL*. (Preprint). ACL Anthology+1


[not bad]
Thanks for noth
sick
Killer battery
Could you please…
Cruaks for killer battery
crash
Sentiment
Not
Negative
Drift Monitor